特别说明

此资料来自百度文库（http://wenku.baidu.com/）

您目前所看到的文档是使用的**抱米花百度文库下载器**所生成

此文档原地址来自

http://wenku.baidu.com/view/3341b7202f60ddccda38a011.html

感谢您的支持

抱米花

http://blog.sina.com.cn/lotusbaob

VISIT…

LANZAROTE

Caliente.COM

FLUENT 6.3 使用手册

B.Case 和 Data 文件格式

本目录描述 Fluent 中 Case 和 Data 文件的内容和格式。
各个章节描述内容根据功能分组：

- 章节 B.2：格式文件和二进制文件的区别。
- 章节 B.3：在 Fluent 中创建网格。
- 章节 B.4：边界条件，材料性质和求解器控制参数。
- 章节 B.3 和章节 B.5：将求解结果导入到其他后处理软件。

Case 和 Date 文件还可能包含其他仅在内部使用的章节。

B.1 基本准则

Fluent 中的 Case 和 Data 文件根据以下基本准则分为几个部分：

- 每个部分都在一对圆括号中并且以一个标志其类型的十进制整数开始。对于格式的二进制文件这个整数是不同的。
- 所有的项目都位于圆括号中。这使得跳过到部分结尾和分解他们都非常容易，还容许在将来输出时加入简单且相容的新项目。
- 各个项目列表的开头信息都位于各对独立的圆括号中，并且项目位于他们自己的圆括号中。

B.2 形成二进制格式文件的规定

对于格式文件的例子在章节 B.3 和 B.4 中给出。对于二进制文件，如果是单精度的二进制数据，在本章节描述的开头标志(如作为结点标志的 10)前加 20；如果是双精度的数据前面加 30(如 2010 或 3010 而不是 10)。二进制文件的结尾在本部分结束参数之前以 End of Binary Section 2010 或 End of Binary Section 3010 为标志。

以时间为代表的二进制数据例子如下所示：

```
(2010 (2 1 2aad 2 3)(
.
.
.
)
End of Binary Section 2010)
```

B.3 网格部分

网格部分存储在 Case 文件中。网格文件是 Case 文件的一个子集，其中仅包含关于网格的部分。目前定义的网格部分包括：

- 注释(见 B.3.1)
- 开头(见 B.3.2)
- 维度(见 B.3.3)
- 结点(见 B.3.4)

- 周期性的阴影面(见 B.3.5)
- 网格单元(B.3.6)
- 网格面(B.3.7)
- 面结构(B.3.8)
- 单元结构(B.3.9)
- 接触面(B.3.10)

每个部分的 ID 号码都以符号和数字两种形式标明。符号的描述在 Fluent Inc 下的 Scheme 源文件中(xfile.scm)或者在一个 C 开头文件中(xfile.h),此文件位于安装目录的如下路径中:/Fluent.Inc/fluent6.x/client/src/xfile.h

B.3.1 注释

Index:	0
Scheme symbol:	`xf-comment`
C macro:	`XF_COMMENT`
Status:	optional

注释部分可以出现在文件的任何位置(除了其他部分的内部)如:

```
(0 "comment text")
```

建议在每个长的部分或者一系列相关的部分前加上一个注释来解释下面的内容。

例如:

```
(0 "Variables:")
(37 (
(relax-mass-flow 1)
(default-coefficient ())
(default-method 0)
))
```

B.3.2 开头

Index:	1
Scheme symbol:	`xf-header`
C macro:	`XF_HEADER`
Status:	optional

开头部分可以出现在文件的任何位置(除了其他部分的内部)下面是一个例子:

```
(1 "TGrid 2.1.1")
```

这个部分的目的是识别写文件的程序入口。虽然它能出现在文件的任何位置，但是一般位于一个文件的开头部分。另外的开头部分标记用来生成文件的其他程序入口。它提供了一种显示一个文件从哪里来以及它怎样被处理的历史机制。

B.3.3 维度

Index: 2
Scheme symbol: xf-dimension
C macro: XF_DIMENSION
Status: optional

网格的维度这样出现：

```
(2 ND)
```

其中 ND 为 2 或者 3。这部分作为对网格是否有一个合适的维度的检查。

B.3.4 节点

Index: 10
Scheme symbol: xf-node
C macro: XF_NODE
Status: required

格式：

```
(10 (zone-id first-index last-index type ND)(
x1 y1 z1
x2 y2 z2
.
.
.
))
```

如果 zone-id 是 0，那么就表示了网格中节点的总数。所以 first-index 是 1，last-index 是以十六进制表示的节点总数，type 等于 1，ND 是网格的维度，并且下面没有坐标(坐标的圆括号也省略了)。

例如：(10 (0 1 2d5 1 2))

如果 zone-id 大于 0，那么它标志着这些节点属于的区域。first-index 和 last-index 是以十六进制表示的区域中节点的标志。每个区域中 last-index 的值必须小于或等于声明部分中的值。type 始终等于 1。ND 是一个可选择项，它表示节点数据的维度，在这里是 2 或者 3。

如果网格是二维的，就像节点数据开头确定的那样，那么每行中只有 x、y 的坐标。

下面是一个二维网格的例子：

```
(10 (1 1 2d5 1 2)(
1.500000e-01 2.500000e-02
1.625000e-01 1.250000e-02
          .
          .
          .
1.750000e-01 0.000000e+00
2.000000e-01 2.500000e-02
1.875000e-01 1.250000e-02
))
```

因为网格的连通性是由代表各个位置的整数组成的，所以使用十六进制可以节省文件空间并且提供更快的文件输入输出速度。开头的标志是十六进制的，这样就可以和网格连通性部分的标志进行匹配。为了保持一致性，zone-id 和 type 也采用十六进制。

B.3.5 周期性的阴影面

Index: 18

Scheme symbol: xf-periodic-face

C macro: XF_PERIODIC_FACE

Status: required only for grids with periodic boundaries

这部分表明在周期性边界上的周期性面对配对。没有周期性边界的网格没有这一部分。这部分的格式如下：

```
(18 (first-index last-index periodic-zone shadow-zone)(
f00 f01
f10 f11
f20 f21
.
.
.
))
```

其中

first-index:列表中成对出现的第一个周期面的指针

last-index:列表中成对出现的最后一个周期面的指针

periodic-zone:周期面所在区域的 ID

shadow-zone:相应的阴影面区域的 ID

这些都是十六进制格式。这部分的指针(f*)涉及各个周期性边界上的面(以十六进制)，这些指针表示网格中面的补偿量。

注意：在这种情况下，first-index 和 last-index 不涉及面的指针，而是涉及列表中成对的周期性面的指针。

例如：

```
(18 (1 2b a c) (
12 1f
13 21
ad 1c2
.
.
.
))
```

B.3.6 网格单元

Index:	**12**
Scheme symbol:	xf-cell
C macro:	XF_CELL
Status:	**required**

网格单元的声明部分与节点的类似：

```
(12 (zone-id first-index last-index type element-type))
```

同样，zone-id 是 0 表明这是对单元总数的声明。如果 last-index 是 0，那么网格中没有单元。当文件仅包含一个网格面时，这一点很有用，可以警告 Fluent 这个网格不能使用。如下例子中，type 的值为 0 而 element-type 没有出现。

例如：

```
(12 (0 1 3e3 0))
```

它表明网格中共有 3e3(十六进制)=995 个单元。声明部分是必需的并且要在正常的网格单元部分之前。正常网格单元部分开头的 element-type 表示该部分中网格单元的类型，如下所示：

Element-type	描述	节点/单元	面/单元
0	混合型		
1	三角形	3	3
2	四面体	4	4
3	四边形	4	4
4	六面体	8	6
5	棱锥体	5	5
6	楔形	6	5
7	多面体	NN	NF

其中 NN 和 NF 取值是不确定的，取决于所确定的多面体单元。

正常的网格单元部分没有主体，但是它们有一个相同格式的开头，其中 first-index 和 last-index 表示特定区域的范围，type 表示单元区域是否是活动的(固体或者液体)。活动区域用 type 等于 1 表示，非活动区域用 type 等于 32 表示。

在 Fluent 的早期版本中，固体和液体区域使用时有区别。这一点现在通过属性决定(如材料类型)。

type 等于 0 表示无用的区域并且会被 Fluent 跳过。如果一个区域是混合类型的(element-type=0),那么会有一个主体部分来列出每一个网格单元的 element-type。例如：

```
(12 (9 1 3d 0 0)(
1 1 1 3 3 1 1 3 1
.
.
.
))
```

这里在单元区域 9 中有 3d(十六进制)=61 个网格单元,其中前三个网格单元是三角形，接下来两个是四边形......等等。

B.3.7 网格面

Index: 13

Scheme symbol: `xf-face`

C macro: `XF_FACE`

Status: required

网格面部分的格式如下：

```
(13 (zone-id first-index last-index bc-type face-type))
```

其中
zone-id:网格面部分的区域 ID
first-index:列表中第一个面的指针
last-index:列表中最后一个面的指针
bc-type:网格面部分描述的边界条件的 ID
face-type:网格面类型的 ID
现在有效的边界条件类型定义如下表所示:

bc-type	描述
2	内部
3	壁面
4	压力-入口，入口-通风孔，通风口-风扇
5	压力出口，排气-风扇，出口-通风口
7	对称面
8	周期性阴影面
9	压力区
10	速度入口
12	周期
14	风扇，多孔介质阶跃面，散热器
20	质量流出口
24	接触面
31	父面(悬挂节点)
36	流出
37	轴

非正形网格交集形成的网格面被放在一个单独的网格面区域中，并且要在 bc-type 上加上 1000(例如 1003 是一个壁面的区域)。
现在有效的网格面类型定义如下表所示:

face-type	描述	节点/面
0	混合型	
2	直线	2
3	三角形	3
4	四边形	4
5	多边形	NN

其中 NN 是不确定的，取决于所确定的多边形面。

zone-id 为 0 表示一个声明部分，其中给出了文件中网格面的总数。这样的一个部分忽略了 bc-type 并且下面没有包含更多信息的主体。

zone-id 非 0 表示一个正常的网格面部分，下面有一个包含网格连续性信息的主体。主体的每一行都描述一个网格面，并且有如下格式：

```
n0 n1 n2 c0 c1
```

其中

n*：定义面的节点(或点)

c*：邻近的网格单元

这是一个具有三角形网格面的三维网格格式。实际的节点数量取决于 face-type。网格单元的顺序很重要，由右手定则决定：如果你以节点的顺序为方向弯曲右手的手指，那么你的大拇指将指向 c1。

对于二维网格，n2 是省略的。C1 取决于 $\hat{r}$ 和 $\hat{k}$ 两个向量的叉乘。向量 $\hat{r}$ 从 n0 延伸到 n1，而向量 $\hat{k}$ 以 n0 为起点，向着观察者的方向指出网格平面。如果你右手沿着 $\hat{r}$ 的方向伸出并且沿着 $\hat{r}$ 和 $\hat{k}$ 之间夹角的方向弯曲你的手指，那么你的大拇指将沿着 $\hat{r} \times \hat{k}$ 的方向指向 c1。

如果网格面区域是混合型的(face-type=0)，那么这部分主体的每一行都以一个组成该面的节点数量参考值开始。其格式如下：

```
x n0 n1 ... nf c0 c1
```

其中

x：网格面的节点(或点)数量

nf：网格面的末节点

所有的网格单元、网格面和节点都有确定的指针。如果一个面在某一边只有一个网格单元，那么 c0 或 c1 为 0。对于仅仅包含一个曲面网格的文件，这两个值都为 0。

对于 Fluent 中不同网格单元类型的面-节点连通性信息，请参考章节 6.1.2。

B.3.8 面结构

Index:	59
Scheme symbol:	`xf-face-tree`
C macro:	`XF_FACE_TREE`
Status:	only for grids with hanging-node adaption

这一部分表示包含悬挂节点的网格面的层级。其格式如下：

```
(59 (face-id0 face-id1 parent-zone-id child-zone-id)
(
 number-of-kids kid-id-0 kid-id-1 ... kid-id-n
 .
 .
 .
))
```

其中
face-id0：该部分中第一个父面的指针
face-id1：该部分中最后一个父面的指针
parent-zone-id：包含父面区域的 ID
child-zone-id：包含子面区域的 ID
number-of-kids：父面包含的子面个数
kid-id-n：子面的 ID
以上这些都是十六进制格式。

B.3.9 单元结构

Index:	58
Scheme symbol:	xf-cell-tree
C macro:	XF_CELL_TREE
Status:	only for grids with hanging-node adaption

这一部分表示包含悬挂节点的网格的单元的层级。其格式如下：

```
(58 (cell-id0 cell-id1 parent-zone-id child-zone-id)
(
 number-of-kids kid-id-0 kid-id-1 ... kid-id-n
 .
 .
 .
))
```

其中
cell-id0：该部分中第一个父单元的指针
cell-id1：该部分中最后一个父单元的指针
parent-zone-id：包含父单元区域的 ID
child-zone-id：包含子单元区域的 ID
number-of-kids：父单元包含的子单元个数
kid-id-n：子单元的 ID
以上这些都是十六进制格式。

B.3.10 接触面

Index: 61

Scheme symbol: xf-face-parents

C macro: XF_FACE_PARENTS

Status: only for grids with non-conformal interfaces

这一部分表示相交面和原始面之间的关系。相交面是通过将两个非正形的表面取交集以及原始面的一部分形成的。每一个子面至少涉及一个父面，该部分的格式如下：

```
(61 (face-id0 face-id1)
(
 parent-id-0 parent-id-1
 .
 .
 .
))
```

其中

Face-id0：该部分第一个子面的指针

Face-id1：该部分最后一个子面的指针

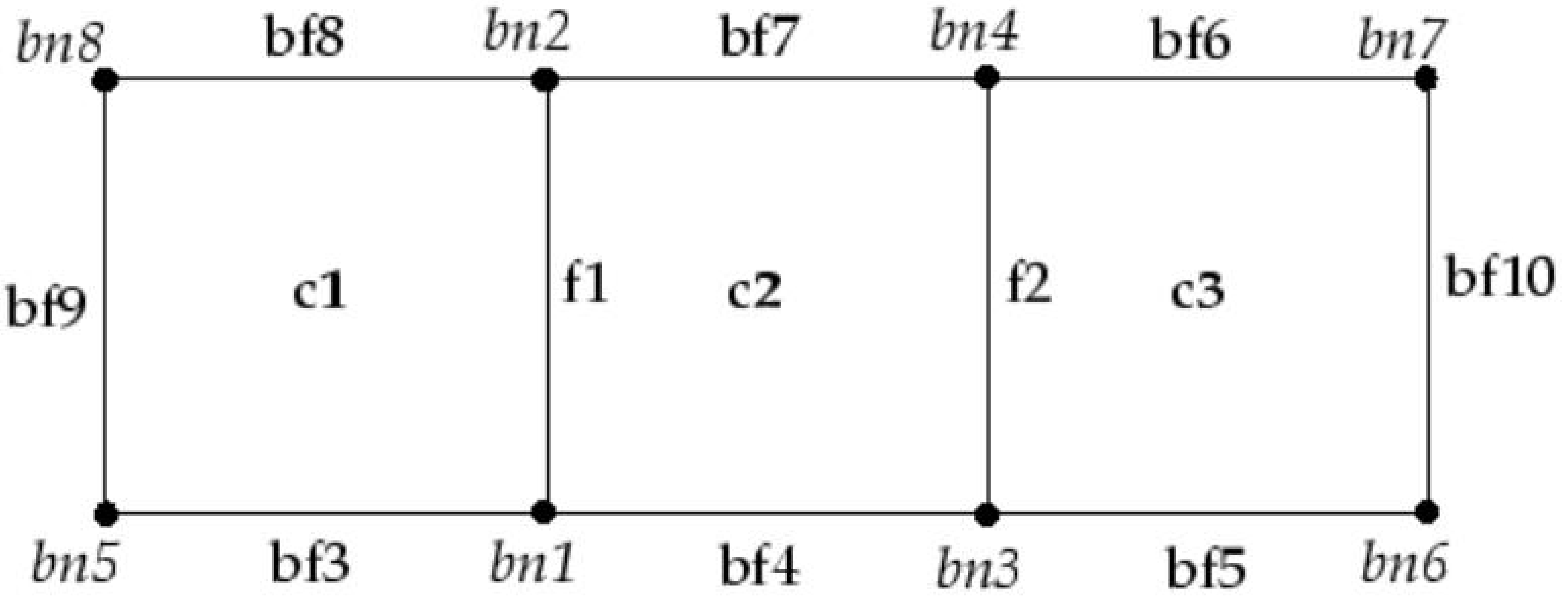
Parent-id-*：父面的指针

以上这些都是十六进制格式。

如果你从 Fluent 中读取一个非正形的网格到 TGrid，TGrid 会跳过该部分，所以它不包含保存该非正形接触面的所有必需信息。当你把网格读回到 Fluent 时，你需要重新生成这个接触面。

实例文件：

例 1：

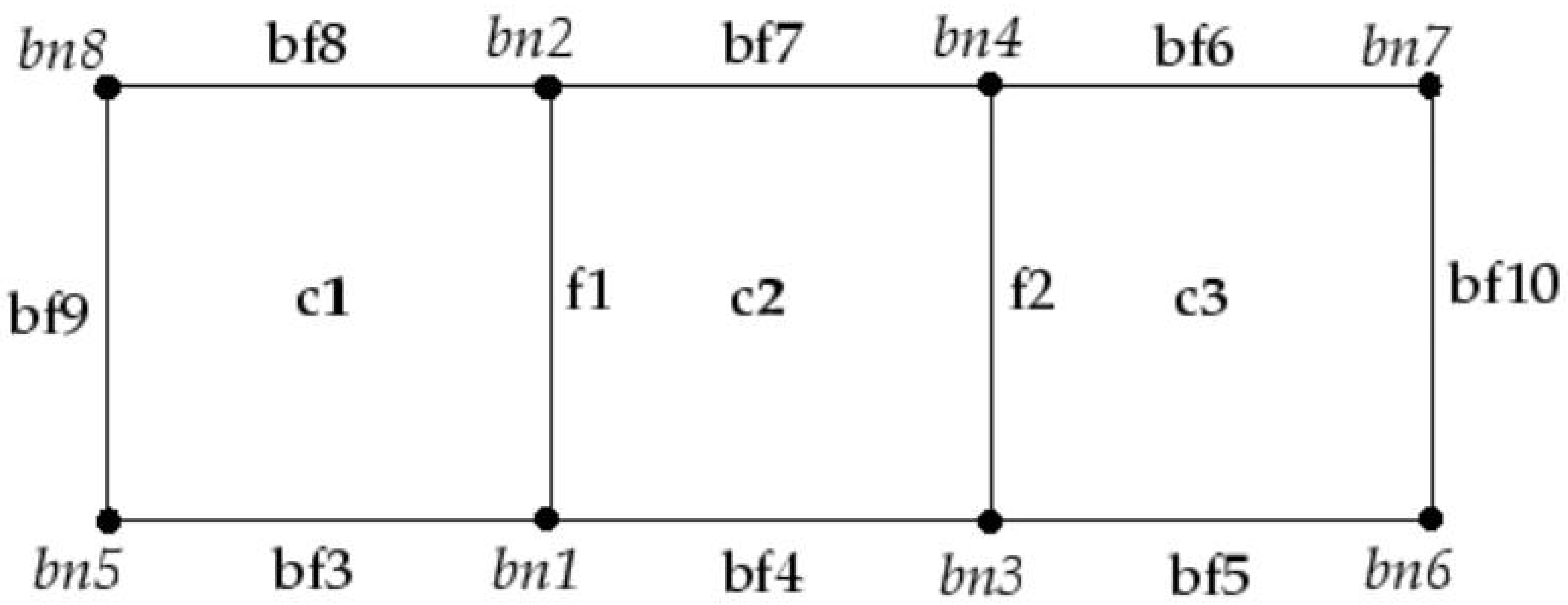


如图所示是一个不包含周期性边界和悬挂节点的简单四边形网格。

该网格可描述如下：

```
(0 "Grid:")

(0 "Dimensions:")
(2 2)

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
2 8 1 0))

(13 (5 9 9 a 2)(
8 5 1 0))

(13 (6 a a 24 2)(
6 7 3 0))

(10 (1 1 8 1 2)
(
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 2.00000000e+00  1.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```

例 2：

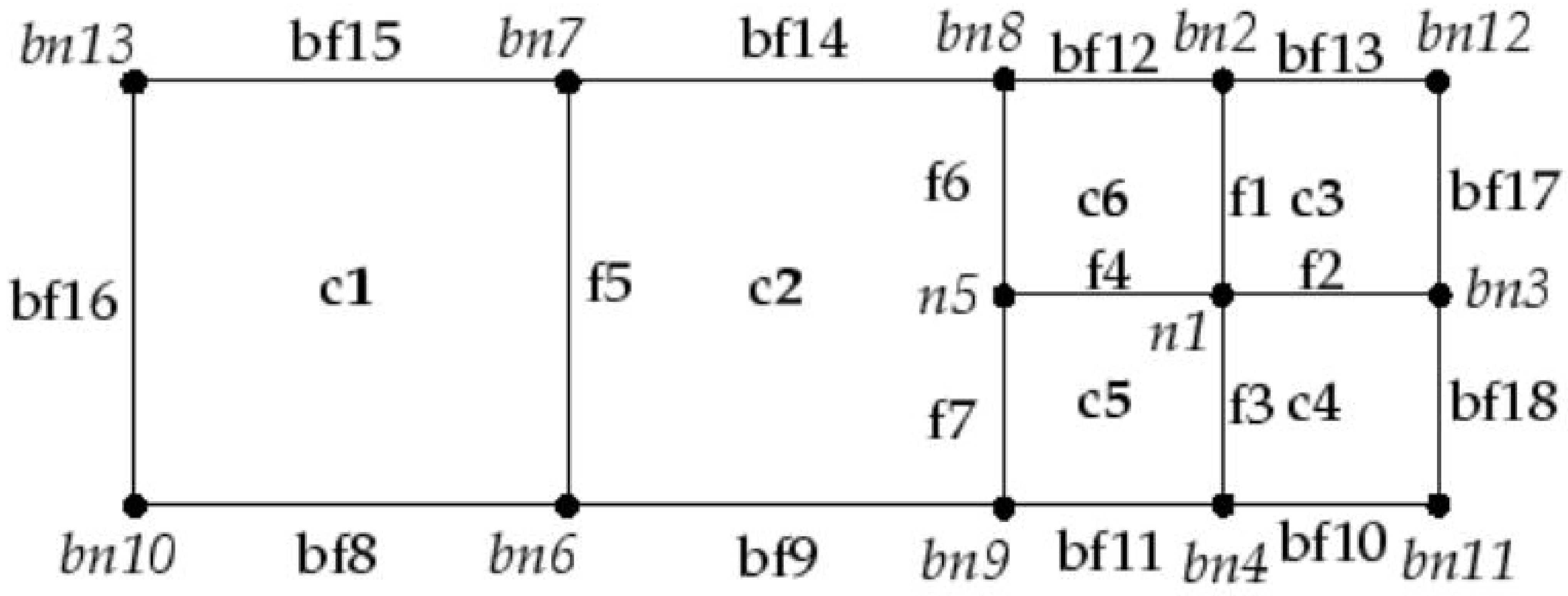


如图所示是一个具有周期性边界但是没有悬挂节点的简单四边形网格。在本例中，bf9 和 bf10 是周期性区域中的网格面。

该网格可描述如下：

```
(0 "Dimensions:")
(2 2)

(0 "Grid:")

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
2 8 1 0))

(13 (5 9 9 c 2)(
8 5 1 0))

(13 (1 a a 8 2)(
6 7 3 0))

(18 (1 1 5 1)(
 9 a))

(10 (1 1 8 1 2)(
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 2.00000000e+00  1.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```

例 3:



如图所示是一个有悬挂节点的简单四边形网格。
该网格可描述如下：

```
(0 "Grid:")

(0 "Dimensions:")
(2 2)

(12 (0 1 7 0))
(13 (0 1 16 0))
(10 (0 1 d 0 2))

(12 (7 1 6 1 3))
(12 (1 7 7 20 3))

(58 (7 7 1 7)(
 4 6 5 4 3))

(13 (2 1 7 2 2)(
1 2 6 3
1 3 3 4
1 4 4 5
1 5 5 6
6 7 1 2
5 8 2 6
9 5 2 5))

(13 (3 8 b 3 2)(
a 6 1 0
6 9 2 0
4 b 4 0
9 4 5 0))

(13 (4 c f 3 2)(
2 8 6 0
c 2 3 0
8 7 2 0
7 d 1 0))

(13 (5 10 10 a 2)(
d a 1 0))

(13 (6 11 12 24 2)(
3 c 3 0
b 3 4 0))

(13 (b 13 13 1f 2)(
c 8 7 0))

(13 (a 14 14 1f 2)(
b c 7 0))

(13 (9 15 15 1f 2)(
9 b 7 0))

(13 (8 16 16 1f 2)(
9 8 2 7))

(59 (13 13 b 4)(
 2 d c))

(59 (14 14 a 6)(
 2 12 11))

(59 (15 15 9 3)(
 2 b a))

(59 (16 16 8 2)(
 2 7 6))

(10 (1 1 d 1 2)
(
 2.50000000e+00  5.00000000e-01
 2.50000000e+00  1.00000000e+00
 3.00000000e+00  5.00000000e-01
 2.50000000e+00  0.00000000e+00
 2.00000000e+00  5.00000000e-01
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```

B.4 其他(非网格)部分

下面的部分存储边界条件、材料性质和求解器控制参数。

B.4.1 区域

Index:	39 or 45
Scheme symbol:	`xf-rp-tv`
C macro:	`XF_RP_TV`
Status:	required

对于网格的每一个区域有一个典型的区域部分。虽然有些网格区域可能没有相对应的区域部分，但是对每个区域不可能有多余一个的区域部分。

区域部分格式如下：

```
(39 (zone-id zone-type zone-name domain-id)(
 (condition1 . value1)
 (condition2 . value2)
 (condition3 . value3)
 .
 .
 .
))
```

网格生成软件和前处理软件仅需要提供开头部分而把条件列表留为空，如：

```
(39 (zone-id zone-type zone-name domain-id)())
```

末尾的空白圆括号是必需的。求解器添加适当的条件，取决于区域的类型。只有当 zone-id、zone-type、zone-name 和 domain-id 都被确定，指针 45 用作为区域部分的指针。然而，如果边界条件是现在的，指针 39 必须被使用，因为以 45 为指针的部分在 zone-id、zone-type、zone-name 和 domain-id 后的任何信息都会被忽略。

这里的 zone-id 是十进制格式。这是与网格部分使用的十六进制对比。zone-type 是以下列表中的一种：

```
axis
exhaust fan
fan
fluid
inlet vent
intake fan
interface
interior
mass-flow-inlet
outlet vent
outflow
periodic
porous-jump
pressure-far-field
pressure-inlet
pressure-outlet
radiator
shadow
solid
symmetry
velocity-inlet
wall
```

其中 Interior、fan、porous-jump 和 radiator 类型只能被分配给网格面在范围里的区域。interior 类型用来给没有单元区域的网格面；其他的用来给那些形成区域内无限薄表面的内部网格面。Fluent 允许 wall 类型分配给在内部或者边界上的网格面。有些区域类型只对特定类型的网格组件有效。例如，单元区域只能被分配如下两种类型之一：

```
fluid
solid
```

上面列出的所有其他类型只能被用于边界区域。zone-name 是一个对区域由用户确定的标签。它必须是一个有效的符号并且没有被引用。

一个有效的 zone-name 的规则如下：

- 第一个字符必须是一个小写字母或者特定的大写字母。
- 所有随后的字母必须是小写字母、特定的大写字母、数字或者特定的符号。

其中特殊的字符如下：

```
! $ % & * / : < = > ? ~ _ ^
```

特定的符号是如下之一：

```
. + -
```

有效的区域名称如：inlet-port/cold!、eggs/easy 和 e=m*c^2。

一些由网格生成软件和前处理软件产生的区域部分例子如下：

```
(39 (1 fluid fuel 1)())

(39 (8 pressure-inlet pressure-inlet-8 2)())

(39 (2 wall wing-skin 3)())

(39 (3 symmetry mid-plane 1)())
```

domain-id 是一个出现在区域名字后面的整数，将边界条件和某个特定的阶段联系起来。

B.4.2 分割

Index: 40
Scheme symbol: xf-partition
C macro: XF_PARTITION
Status: only for partitioned grids

该部分表示每个网格单元的分区。其格式如下：

```
(40 (zone-id first-index last-index partition-count)(
p1
p2
p3
.
.
.
pn
))
```

其中
p1：ID 为 first-index 的网格单元的分区
p2：ID 为 first-index+1 的网格单元的分区......等等
pn：ID 为 last-index 的网格单元的分区
partition-count：分区的总数量
分区 ID 必须在 0 和 partition-count 之间。

B.5 数据部分

如下部分存储迭代、残差和数据区的值。

B.5.1 网格大小

Index: 33

Scheme symbol: `xf-grid-size`

C macro: `XF_GRID_SIZE`

Status: optional

该部分表示符合文件数据的网格中单元、面和节点的数量。该信息用来核查数据和网格是否相互匹配。格式如下：

```
(33 (n-elements n-faces n-nodes))
```

其中整数为十进制格式。

B.5.2 数据区

Index: 300

Scheme symbol: `xf-rf-seg-data`

C macro: `XF_RF_SEG_DATA`

Status: required

该部分列出单元或面区域的一系列变量。数据以 case 文件中单元或面的相同顺序存储。对于每个面或者单元写入单独的变量部分来存储。格式如下：

```
(300 (sub-section-id zone-id size n-time-levels
  n-phases first-id last-id)
( data for cell or face with id = first-id
  data-for-cell-or-face with id = first-id+1
  ..
  data-for-cell-or-face with id = last-id
))
```

其中 sub-section-id 是一个用来确定变量类型的十进制整数(如 1 表示压力，2 表示速度)。这些的完整列表在一个开头文件中(xfile.h)，它位于安装目录下的如下路径：/Fluent.Inc/fluent6.x/client/src/xfile.h

其中

zone-id：单元或者面区域的 ID

size：变量向量的长度

zone-id 与 case 文件中使用的 ID 相匹配。size 是 1 为标量，2 或 3 为向量，等于为每个类型定义的变量的种类数量。n-time-levels 和 n-phases 现在没有使用。

一个定常、单相的二维问题网格单元区域速度场的数据文件部分如下：

```
(300 (2 16 2 0 0 17 100)
(8.08462024e-01  8.11823010e-02
 8.78750622e-01  3.15509699e-02
 1.06139672e+00 -3.74040119e-02
 ...
 1.33301604e+00 -5.04243895e-02
 6.21703446e-01 -2.46118382e-02
 4.41687912e-01 -1.27046436e-01
 1.03528820e-01 -1.01711005e-01
))
```

数据文件中列出的变量取决于写文件时运行的模型。当数据文件被读取时，基于当前模型设置的求解器需要的变量如果不在数据文件中，那么他们就被设置为默认值。数据文件中出现的任何额外的按照当前模型设置不相关的变量都被忽略。

B.5.3 残差

Index: 301

Scheme symbol: xf-rf-seg-residuals

C macro: XF_RF_SEG_RESIDUALS

Status: optional

该部分列出每次迭代中对于特定的数据区变量的残差值：

```
(301 (n residual-section-id size)(
r1
r2
.
.
.
rn
))
```

其中

n：残差的个数

size：变量向量的长度

residual-section-id：标识方程的一个十进制整数

size 是 1 为标量，2 或 3 为向量，等于为每个类型定义的变量的种类数量。根据安装目录下的头文件(xfile.h)定义的常数 C，Residual-section-id 标识该部分中残差存储所对应的方程。

数据文件中列出的残差所对应的方程取决于写文件时运行的模型。对于一个当前的方程，如果数据文件中的历史残差丢失，那么就会被初始化为 0。